COMMUNITY
INNOVATION
WORKSPACE

Wifi - In5-Tech [Code - WelcomeToIn5]

About the Assembly

- A smart lab based out of In5 since Dec 2014
- Over 200 free workshops done
- ASSEMBLY: HACK - Embedded systems, iOT and hardware
- ASSEMBLY: CODE - Software projects - APIs, frameworks, apps
- Age range: 16-60 - students, professionals, entrepreneurs
- Focus on smart technology and practical applications
- Forum: members.theassembly.ae

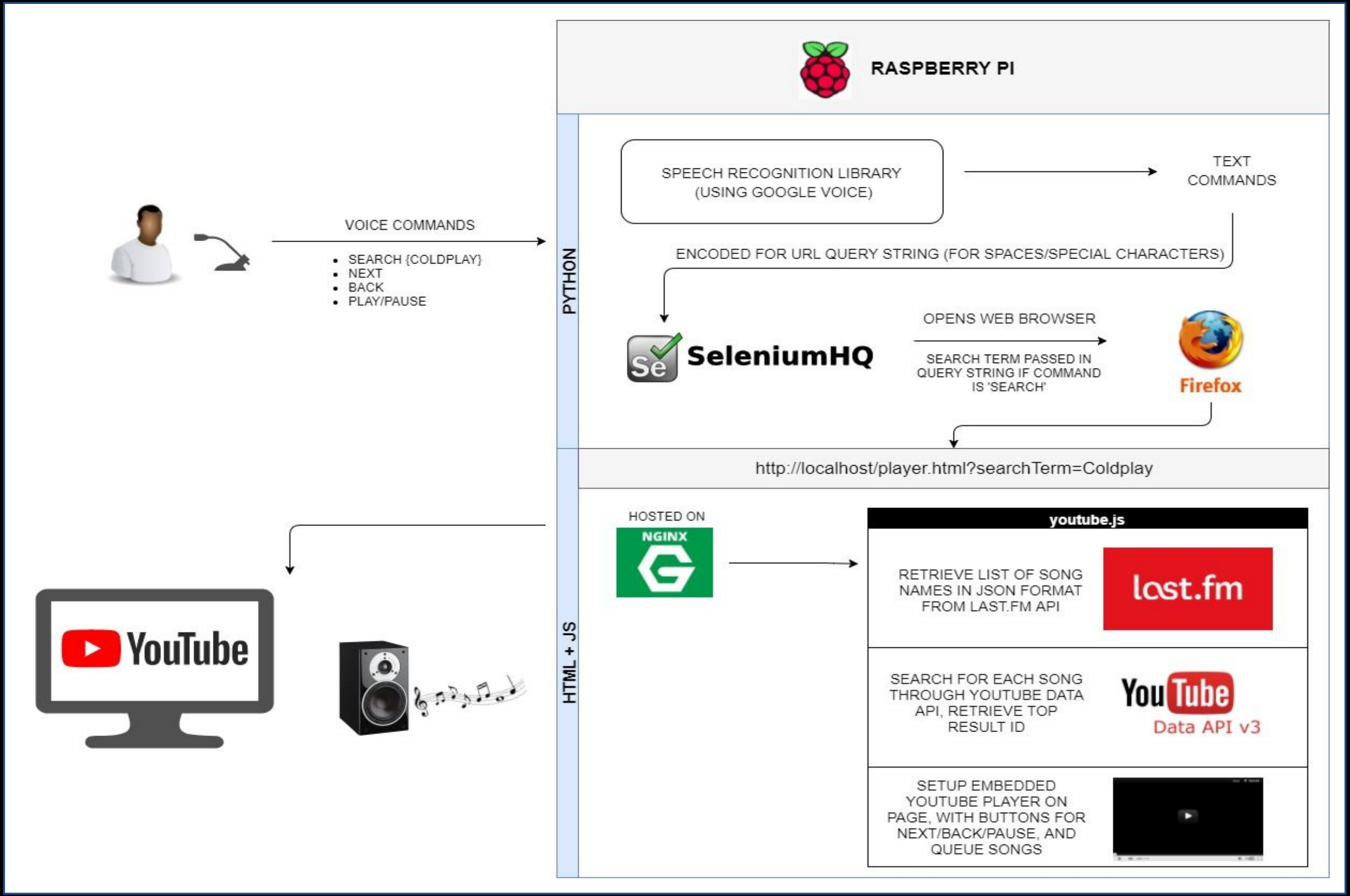
TAG US ON SOCIAL MEDIA!

FACEBOOK - The Assembly (@MakeSmartThings)
TWITTER - @MakeSmartThings
INSTAGRAM - @MakeSmartThings
YOUTUBE - The Assembly

Overview

OBJECTIVE: Create a voice controlled YouTube playlist generator app that runs on a Raspberry Pi.

The user inputs artist name or keywords using the microphone; the app searches for and retrieves songs based on this criteria and then queues them for instant playback as an fully navigable, embedded YouTube playlist on a web page.



RASPBERRY PI

PYTHON

VOICE COMMANDS

- SEARCH {COLDPLAY}
- NEXT
- BACK
- PLAY/PAUSE

SPEECH RECOGNITION LIBRARY (USING GOOGLE VOICE)

TEXT COMMANDS

ENCODED FOR URL QUERY STRING (FOR SPACES/SPECIAL CHARACTERS)

SeleniumHQ

OPENS WEB BROWSER

SEARCH TERM PASSED IN QUERY STRING IF COMMAND IS 'SEARCH'

Firefox

http://localhost/player.html?searchTerm=Coldplay

HTML + JS

HOSTED ON

NGINX

youtube.js

RETRIEVE LIST OF SONG NAMES IN JSON FORMAT FROM LAST.FM API

last.fm

SEARCH FOR EACH SONG THROUGH YOUTUBE DATA API, RETRIEVE TOP RESULT ID

YouTube Data API v3

SETUP EMBEDDED YOUTUBE PLAYER ON PAGE, WITH BUTTONS FOR NEXT/BACK/PAUSE, AND QUEUE SONGS

YouTube

Getting started

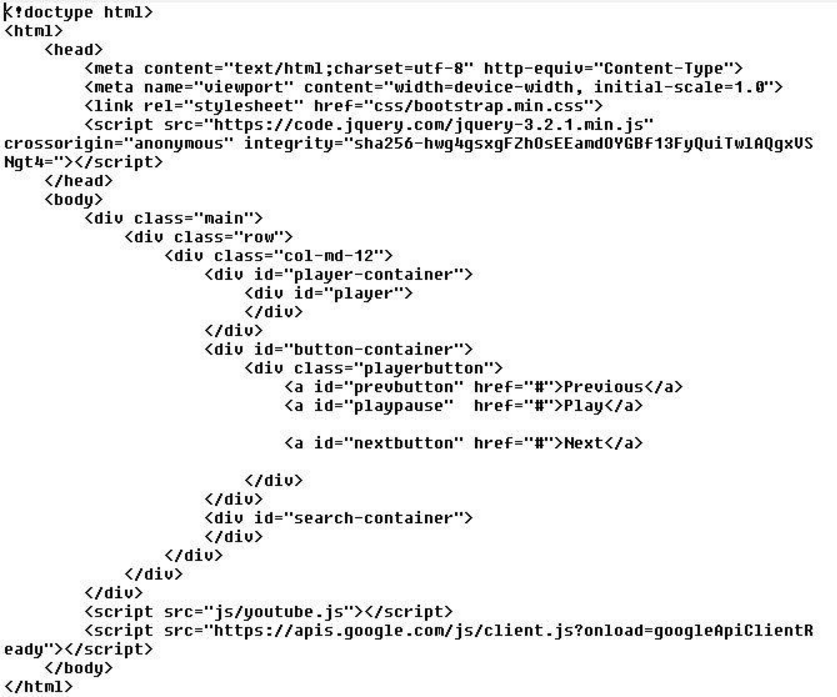
1. Github with all the code: https://github.com/The-Assembly/YouTubeSmartSpeaker
2. Pastebin: https://pastebin.com/t4idcsMQ
3. APIs you need : LastFM API + YouTube Data v3 API
4. Ask for help from the Assembly team if you get stuck with anything!
5. Acknowledgement: Javascript code modified from work done on **Mixbla.st** project

HTML + Javascript

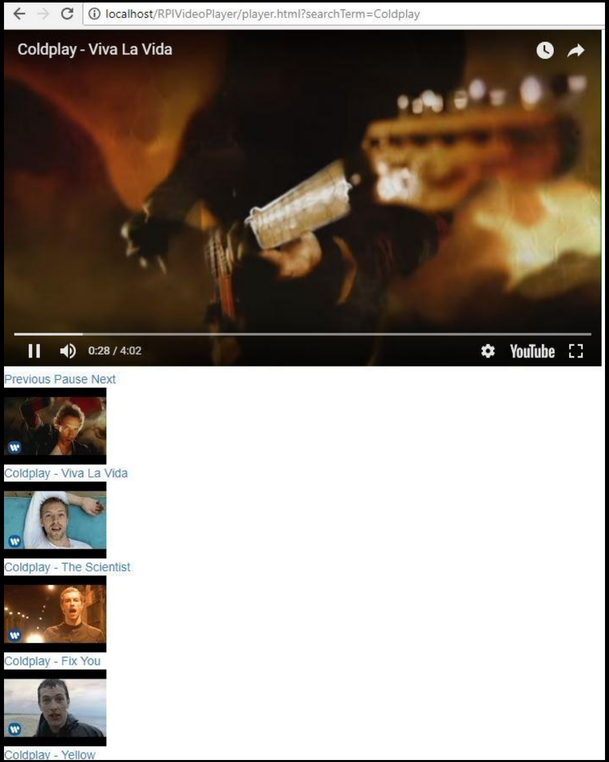
Building the player webpage

- Page receives search term in query string and creates embedded playlist based on that criteria. Playback on page load.
- The page runs on a local webserver (**NGINX**) on the Raspberry Pi
- Code for this portion on Github in directory **RPIVideoPlayer**
  - player.html
  - js\youtube.js
  - css\bootstrap.min.css
- Copy to html folder used by NGINX (usually **/var/www/html**)

player.html



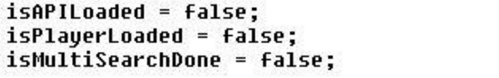
```
<!doctype html>
<html>
    <head>
        <meta content="text/html;charset=utf-8" http-equiv="Content-Type">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <link rel="stylesheet" href="css/bootstrap.min.css">
        <script src="https://code.jquery.com/jquery-3.2.1.min.js"
crossorigin="anonymous" integrity="sha256-hwg4gsxgFZhOsEEamdOYGBf13FyQuiTwlAQgxVS
Ngt4="></script>
    </head>
    <body>
        <div class="main">
            <div class="row">
                <div class="col-md-12">
                    <div id="player-container">
                        <div id="player">
                        </div>
                    </div>
                    <div id="button-container">
                        <div class="playerbutton">
                            <a id="prevbutton" href="#">Previous</a>
                            <a id="playpause"  href="#">Play</a>

                            <a id="nextbutton" href="#">Next</a>

                        </div>
                    </div>
                    <div id="search-container">
                    </div>
                </div>
            </div>
        </div>
        <script src="js/youtube.js"></script>
        <script src="https://apis.google.com/js/client.js?onload=googleApiClientR
eady"></script>
    </body>
</html>
```



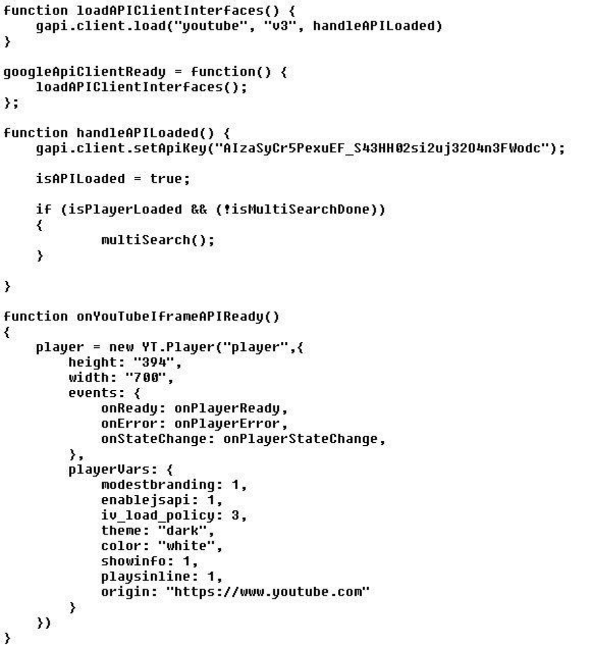
localhost/RPIVideoPlayer/player.html?searchTerm=Coldplay

JS: Loading the YouTube apis

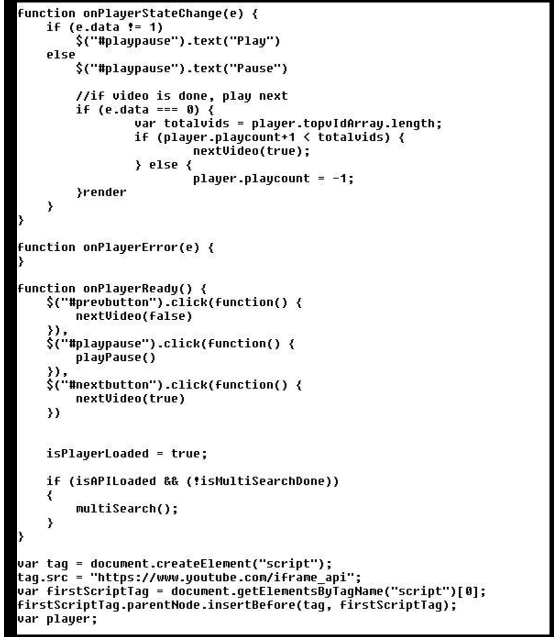
- YouTube Data API and embedded YouTube player iFrame need to BOTH be loaded before starting *multisearch*



```
isAPILoaded = false;
isPlayerLoaded = false;
isMultiSearchDone = false;
```



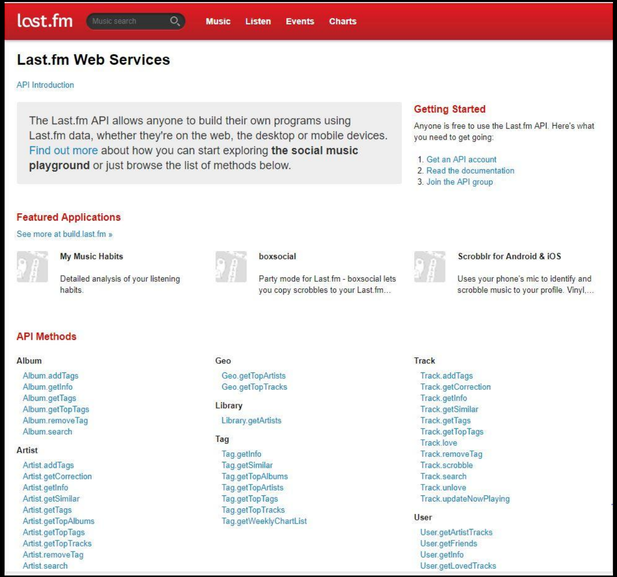
```
function loadAPIClientInterfaces() {
    gapi.client.load("youtube", "v3", handleAPILoaded)
}

googleApiClientReady = function() {
    loadAPIClientInterfaces();
};

function handleAPILoaded() {
    gapi.client.setApiKey("AIzaSyCr5PexuEF_S43HH02si2uj3204n3FWodc");

    isAPILoaded = true;

    if (isPlayerLoaded && (!isMultiSearchDone))
    {
            multiSearch();
    }

}

function onYouTubeIframeAPIReady()
{
    player = new YT.Player("player",{
        height: "394",
        width: "700",
        events: {
            onReady: onPlayerReady,
            onError: onPlayerError,
            onStateChange: onPlayerStateChange,
        },
        playerVars: {
            modestbranding: 1,
            enablejsapi: 1,
            iv_load_policy: 3,
            theme: "dark",
            color: "white",
            showinfo: 1,
            playsinline: 1,
            origin: "https://www.youtube.com"
        }
    })
}
```



```
function onPlayerStateChange(e) {
    if (e.data != 1)
        $("#playpause").text("Play")
    else
        $("#playpause").text("Pause")

        //if video is done, play next
        if (e.data === 0) {
                var totalvids = player.topvIdArray.length;
                if (player.playcount+1 < totalvids) {
                        nextVideo(true);
                } else {
                        player.playcount = -1;
        }render
    }
}

function onPlayerError(e) {
}

function onPlayerReady() {
    $("#prevbutton").click(function() {
        nextVideo(false)
    }),
    $("#playpause").click(function() {
        playPause()
    }),
    $("#nextbutton").click(function() {
        nextVideo(true)
    })


    isPlayerLoaded = true;

    if (isAPILoaded && (!isMultiSearchDone))
    {
        multiSearch();
    }
}

var tag = document.createElement("script");
tag.src = "https://www.youtube.com/iframe_api";
var firstScriptTag = document.getElementsByTagName("script")[0];
firstScriptTag.parentNode.insertBefore(tag, firstScriptTag);
var player;
```

JS: Retrieve titles from LastFM

- https://www.last.fm/api/show/artist.getTopTracks



Last.fm Web Services

API Introduction

The Last.fm API allows anyone to build their own programs using Last.fm data, whether they're on the web, the desktop or mobile devices. Find out more about how you can start exploring **the social music playground** or just browse the list of methods below.

Getting Started

Anyone is free to use the Last.fm API. Here's what you need to get going:

1. Get an API account
2. Read the documentation
3. Join the API group

Featured Applications

See more at build.last.fm »

My Music Habits

Detailed analysis of your listening habits.

boxsocial

Party mode for Last.fm - boxsocial lets you copy scrobbles to your Last.fm...

Scrobblr for Android & iOS

Uses your phone's mic to identify and scrobble music to your profile. Vinyl,...

API Methods

Album

Album.addTags
Album.getInfo
Album.getTags
Album.getTopTags
Album.removeTag
Album.search

Artist

Artist.addTags
Artist.getCorrection
Artist.getInfo
Artist.getSimilar
Artist.getTags
Artist.getTopAlbums
Artist.getTopTags
Artist.getTopTracks
Artist.removeTag
Artist.search

Geo

Geo.getTopArtists
Geo.getTopTracks

Library

Library.getArtists

Tag

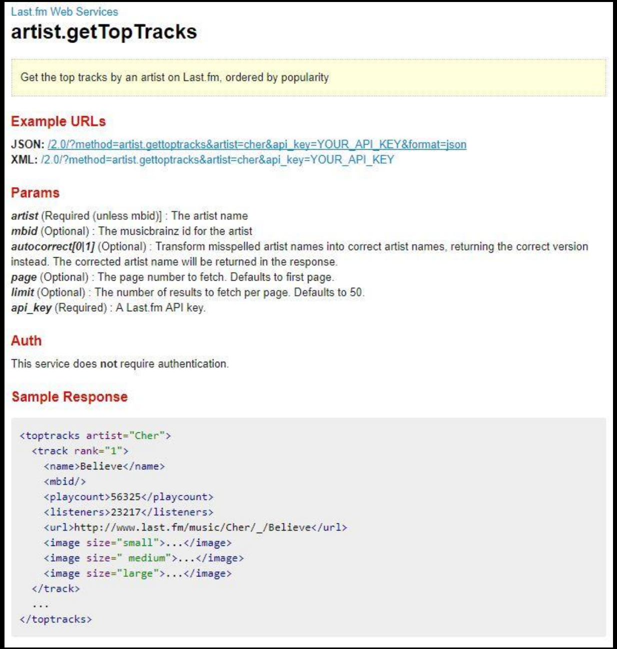
Tag.getInfo
Tag.getSimilar
Tag.getTopAlbums
Tag.getTopArtists
Tag.getTopTags
Tag.getTopTracks
Tag.getWeeklyChartList

Track

Track.addTags
Track.getCorrection
Track.getInfo
Track.getSimilar
Track.getTags
Track.getTopTags
Track.love
Track.removeTag
Track.scrobble
Track.search
Track.unlove
Track.updateNowPlaying

User

User.getArtistTracks
User.getFriends
User.getInfo
User.getLovedTracks



Last.fm Web Services

artist.getTopTracks

Get the top tracks by an artist on Last.fm, ordered by popularity

Example URLs

JSON: /2.0/?method=artist.gettoptracks&artist=cher&api_key=YOUR_API_KEY&format=json
XML: /2.0/?method=artist.gettoptracks&artist=cher&api_key=YOUR_API_KEY

Params

artist (Required (unless mbid)] : The artist name
mbid (Optional) : The musicbrainz id for the artist
autocorrect[0|1] (Optional) : Transform misspelled artist names into correct artist names, returning the correct version instead. The corrected artist name will be returned in the response.
page (Optional) : The page number to fetch. Defaults to first page.
limit (Optional) : The number of results to fetch per page. Defaults to 50.
api_key (Required) : A Last.fm API key.

Auth

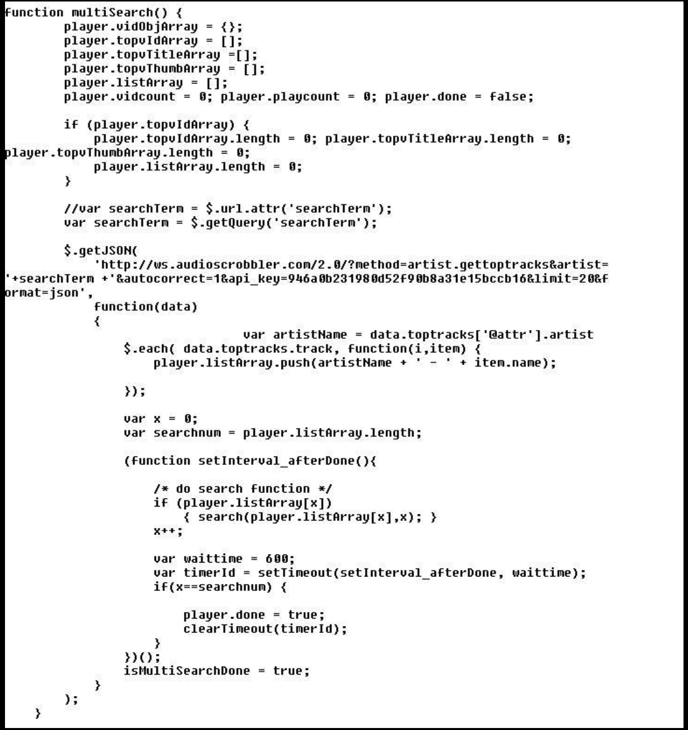
This service does **not** require authentication.

Sample Response

```
<toptracks artist="Cher">
  <track rank="1">
    <name>Believe</name>
    <mbid/>
    <playcount>56325</playcount>
    <listeners>23217</listeners>
    <url>http://www.last.fm/music/Cher/_/Believe</url>
    <image size="small">...</image>
    <image size=" medium">...</image>
    <image size="large">...</image>
  </track>
  ...
</toptracks>
```

JS: Retrieve titles from LastFM

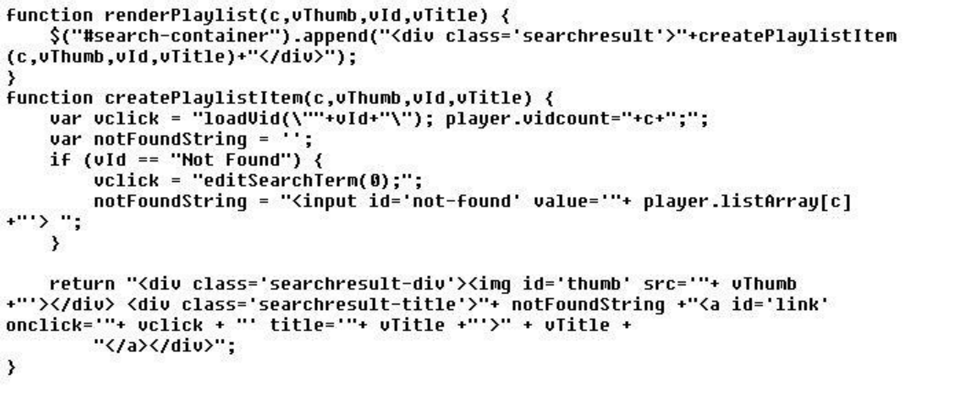
- *multiSearch* sends AJAX web call to **LastFM API** with search term as parameter
- This example uses **artist.gettoptracks** but many other API methods available
- Each item in response list (ie; each song title with artist prefixed) is passed as a parameter to the *search* function



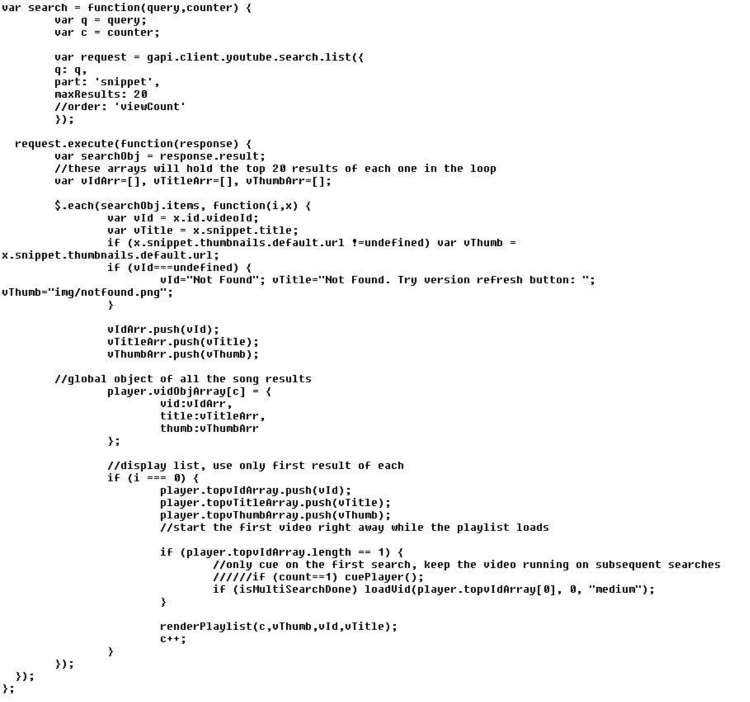
```
function multiSearch() {
        player.vidObjArray = {};
        player.topvIdArray = [];
        player.topvTitleArray =[];
        player.topvThumbArray = [];
        player.listArray = [];
        player.vidcount = 0; player.playcount = 0; player.done = false;

        if (player.topvIdArray) {
            player.topvIdArray.length = 0; player.topvTitleArray.length = 0;
player.topvThumbArray.length = 0;
            player.listArray.length = 0;
        }

        //var searchTerm = $.url.attr('searchTerm');
        var searchTerm = $.getQuery('searchTerm');

        $.getJSON(
            'http://ws.audioscrobbler.com/2.0/?method=artist.gettoptracks&artist=
'+searchTerm +'&autocorrect=1&api_key=946a0b231980d52f90b8a31e15bccb16&limit=20&f
ormat=json',
            function(data)
            {
                                var artistName = data.toptracks['@attr'].artist
                $.each( data.toptracks.track, function(i,item) {
                    player.listArray.push(artistName + ' - ' + item.name);

                });

                var x = 0;
                var searchnum = player.listArray.length;

                (function setInterval_afterDone(){

                    /* do search function */
                    if (player.listArray[x])
                        { search(player.listArray[x],x); }
                    x++;

                    var waittime = 600;
                    var timerId = setTimeout(setInterval_afterDone, waittime);
                    if(x==searchnum) {

                        player.done = true;
                        clearTimeout(timerId);
                    }
                })();
                isMultiSearchDone = true;
            }
        );
    }
```

JS: Search YouTube Data API

- The *search* function finds the top result on YouTube for each song title
- Each result is rendered as a playlist item with title name + screenshot, and queued for playback

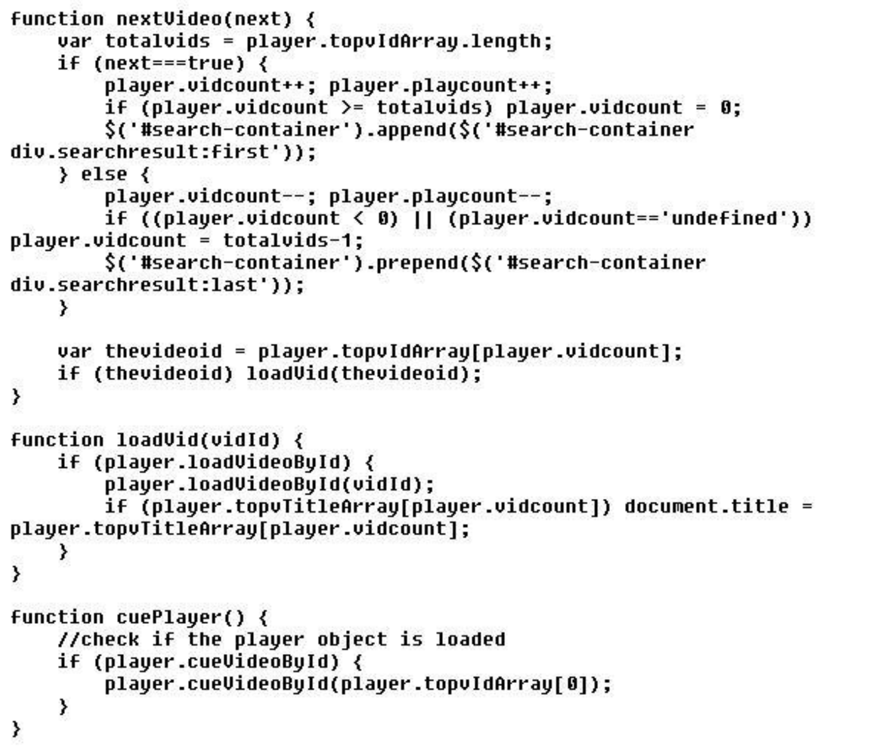


```
function renderPlaylist(c,vThumb,vId,vTitle) {
    $("#search-container").append("<div class='searchresult'>"+createPlaylistItem
(c,vThumb,vId,vTitle)+"</div>");
}
function createPlaylistItem(c,vThumb,vId,vTitle) {
    var vclick = "loadVid(\""+vId+"\"); player.vidcount="+c+";";
    var notFoundString = '';
    if (vId == "Not Found") {
        vclick = "editSearchTerm(0);";
        notFoundString = "<input id='not-found' value='"+ player.listArray[c]
+"'> ";
    }

    return "<div class='searchresult-div'><img id='thumb' src='"+ vThumb
+"'></div> <div class='searchresult-title'>"+ notFoundString +"<a id='link'
onclick='"+ vclick + "' title='"+ vTitle +"'>" + vTitle +
        "</a></div>";
}
```



```
var search = function(query,counter) {
        var q = query;
        var c = counter;

        var request = gapi.client.youtube.search.list({
        q: q,
        part: 'snippet',
        maxResults: 20
        //order: 'viewCount'
        });

  request.execute(function(response) {
        var searchObj = response.result;
        //these arrays will hold the top 20 results of each one in the loop
        var vIdArr=[], vTitleArr=[], vThumbArr=[];

        $.each(searchObj.items, function(i,x) {
                var vId = x.id.videoId;
                var vTitle = x.snippet.title;
                if (x.snippet.thumbnails.default.url !=undefined) var vThumb =
x.snippet.thumbnails.default.url;
                if (vId===undefined) {
                        vId="Not Found"; vTitle="Not Found. Try version refresh button: ";
vThumb="img/notfound.png";
                }

                vIdArr.push(vId);
                vTitleArr.push(vTitle);
                vThumbArr.push(vThumb);

        //global object of all the song results
                player.vidObjArray[c] = {
                        vid:vIdArr,
                        title:vTitleArr,
                        thumb:vThumbArr
                };

                //display list, use only first result of each
                if (i === 0) {
                        player.topvIdArray.push(vId);
                        player.topvTitleArray.push(vTitle);
                        player.topvThumbArray.push(vThumb);
                        //start the first video right away while the playlist loads

                        if (player.topvIdArray.length == 1) {
                                //only cue on the first search, keep the video running on subsequent searches
                                //////if (count==1) cuePlayer();
                                if (isMultiSearchDone) loadVid(player.topvIdArray[0], 0, "medium");
                        }

                        renderPlaylist(c,vThumb,vId,vTitle);
                        c++;
                }
        });
  });
};
```

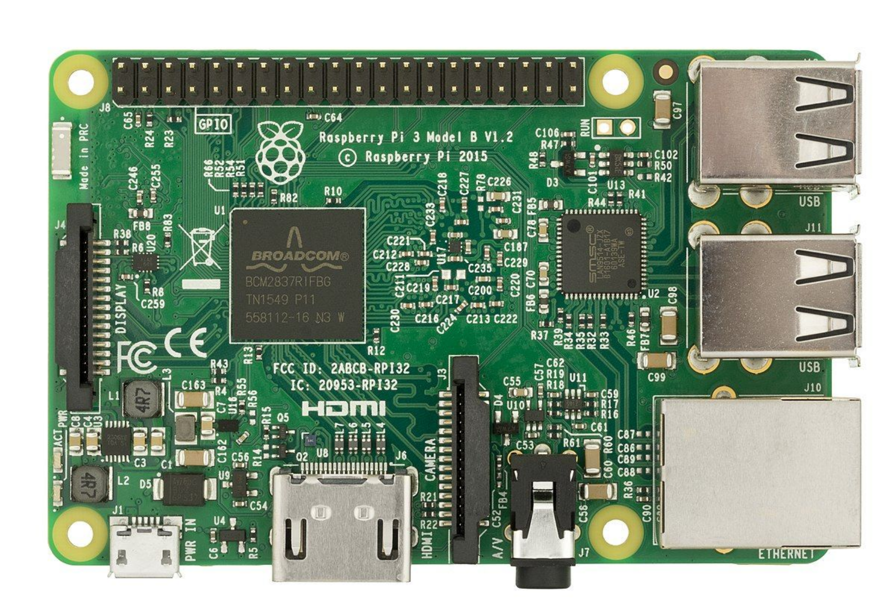
JS: Configure player behavior



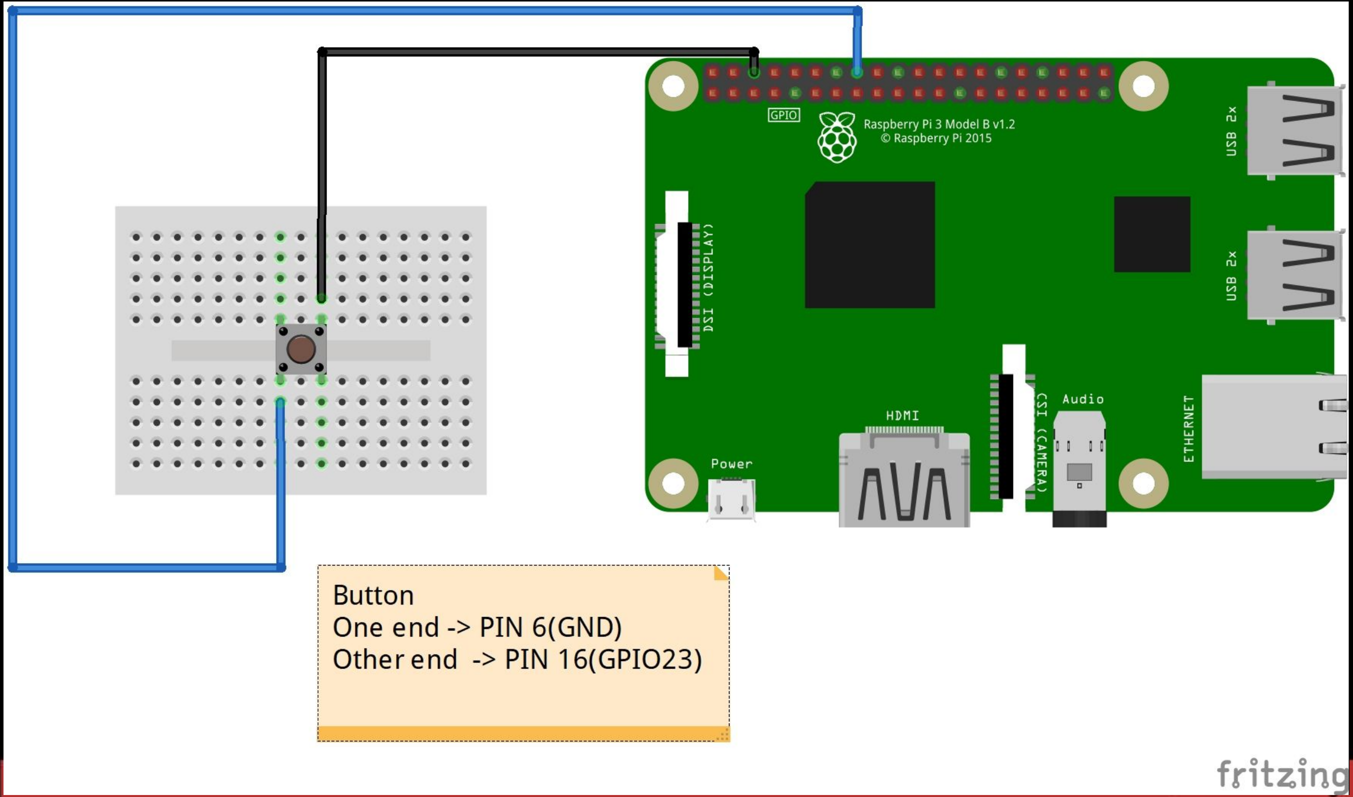
```
function nextVideo(next) {
    var totalvids = player.topvIdArray.length;
    if (next===true) {
        player.vidcount++; player.playcount++;
        if (player.vidcount >= totalvids) player.vidcount = 0;
        $('#search-container').append($('#search-container
div.searchresult:first'));
    } else {
        player.vidcount--; player.playcount--;
        if ((player.vidcount < 0) || (player.vidcount=='undefined'))
player.vidcount = totalvids-1;
        $('#search-container').prepend($('#search-container
div.searchresult:last'));
    }

    var thevideoid = player.topvIdArray[player.vidcount];
    if (thevideoid) loadVid(thevideoid);
}

function loadVid(vidId) {
    if (player.loadVideoById) {
        player.loadVideoById(vidId);
        if (player.topvTitleArray[player.vidcount]) document.title =
player.topvTitleArray[player.vidcount];
    }
}

function cuePlayer() {
    //check if the player object is loaded
    if (player.cueVideoById) {
        player.cueVideoById(player.topvIdArray[0]);
    }
}
```

Hardware

Raspberry Pi



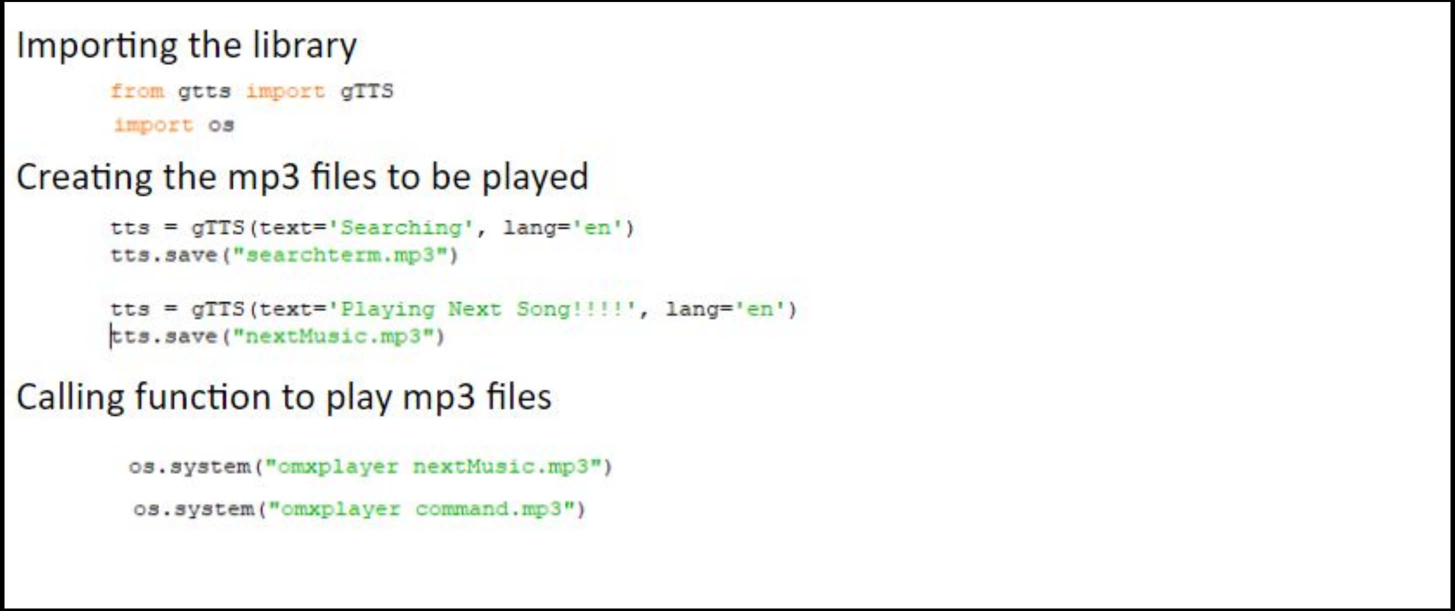
The Circuit Diagram



Button
One end -> PIN 6(GND)
Other end -> PIN 16(GPIO23)

Python Scripting

Generating MP3’s & Google Speech to Text (gTTS)



Importing the library

```
from gtts import gTTS
import os
```

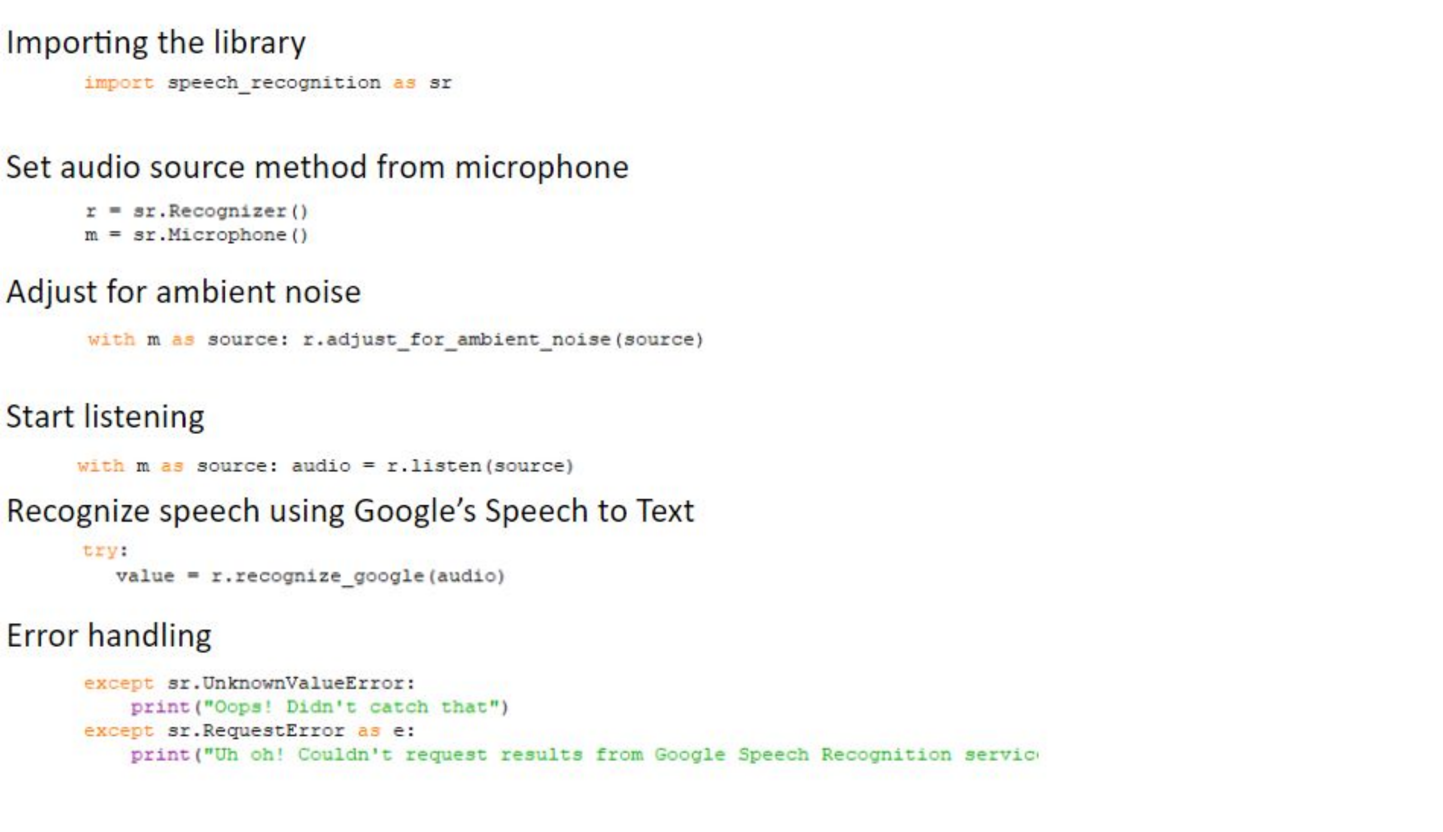
Creating the mp3 files to be played

```
tts = gTTS(text='Searching', lang='en')
tts.save("searchterm.mp3")

tts = gTTS(text='Playing Next Song!!!!', lang='en')
tts.save("nextMusic.mp3")
```

Calling function to play mp3 files

```
os.system("omxplayer nextMusic.mp3")

os.system("omxplayer command.mp3")
```

Speech_Recognition (Speech to text library)



Importing the library

```
import speech_recognition as sr
```

Set audio source method from microphone

```
r = sr.Recognizer()
m = sr.Microphone()
```

Adjust for ambient noise

```
with m as source: r.adjust_for_ambient_noise(source)
```

Start listening

```
with m as source: audio = r.listen(source)
```

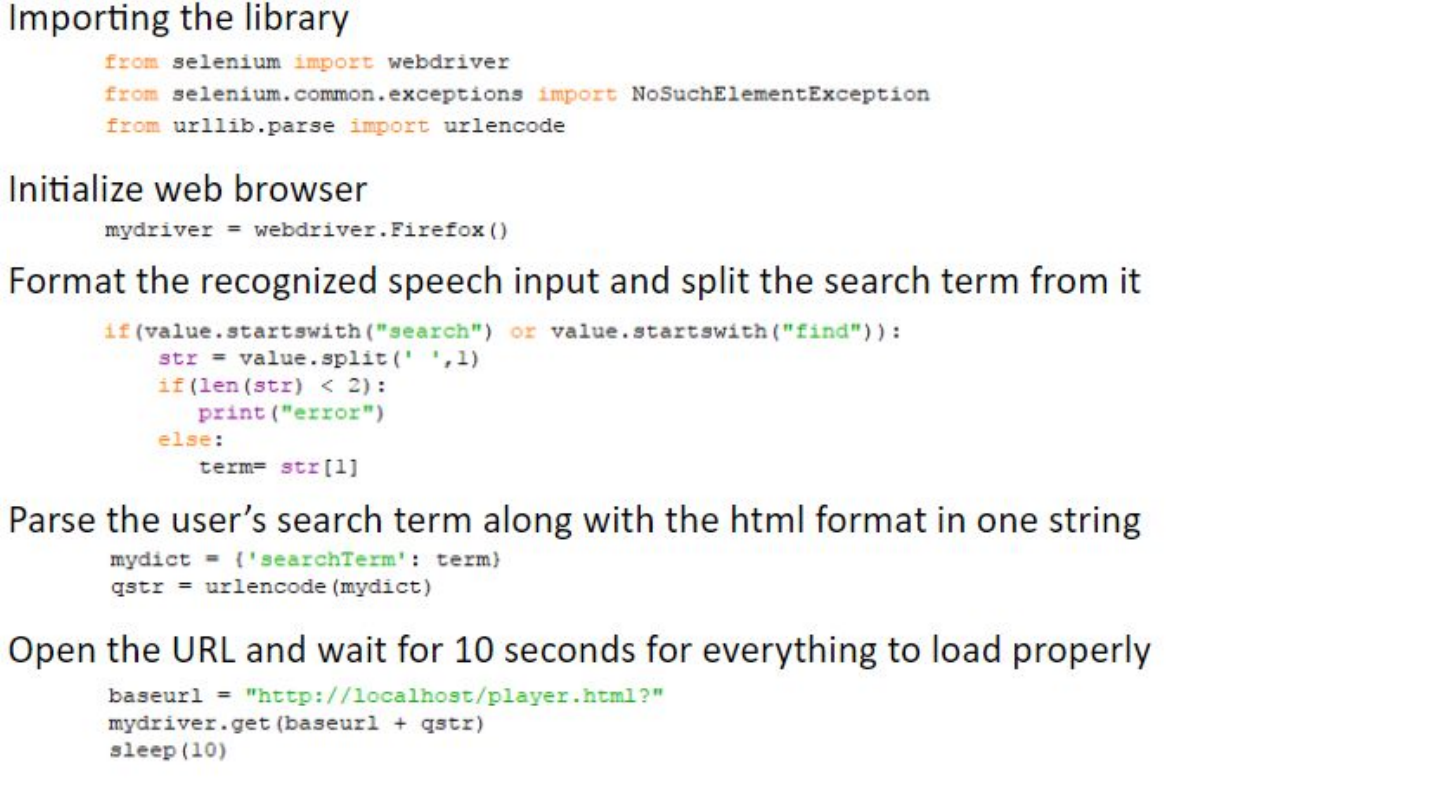
Recognize speech using Google’s Speech to Text

```
try:
   value = r.recognize_google(audio)
```

Error handling

```
except sr.UnknownValueError:
    print("Oops! Didn't catch that")
except sr.RequestError as e:
    print("Uh oh! Couldn't request results from Google Speech Recognition servic
```

Selenium Library to control webpage & URL Parser



Importing the library

```
from selenium import webdriver
from selenium.common.exceptions import NoSuchElementException
from urllib.parse import urlencode
```

Initialize web browser

```
mydriver = webdriver.Firefox()
```

Format the recognized speech input and split the search term from it

```
if(value.startswith("search") or value.startswith("find")):
    str = value.split(' ',1)
    if(len(str) < 2):
       print("error")
    else:
       term= str[1]
```

Parse the user’s search term along with the html format in one string

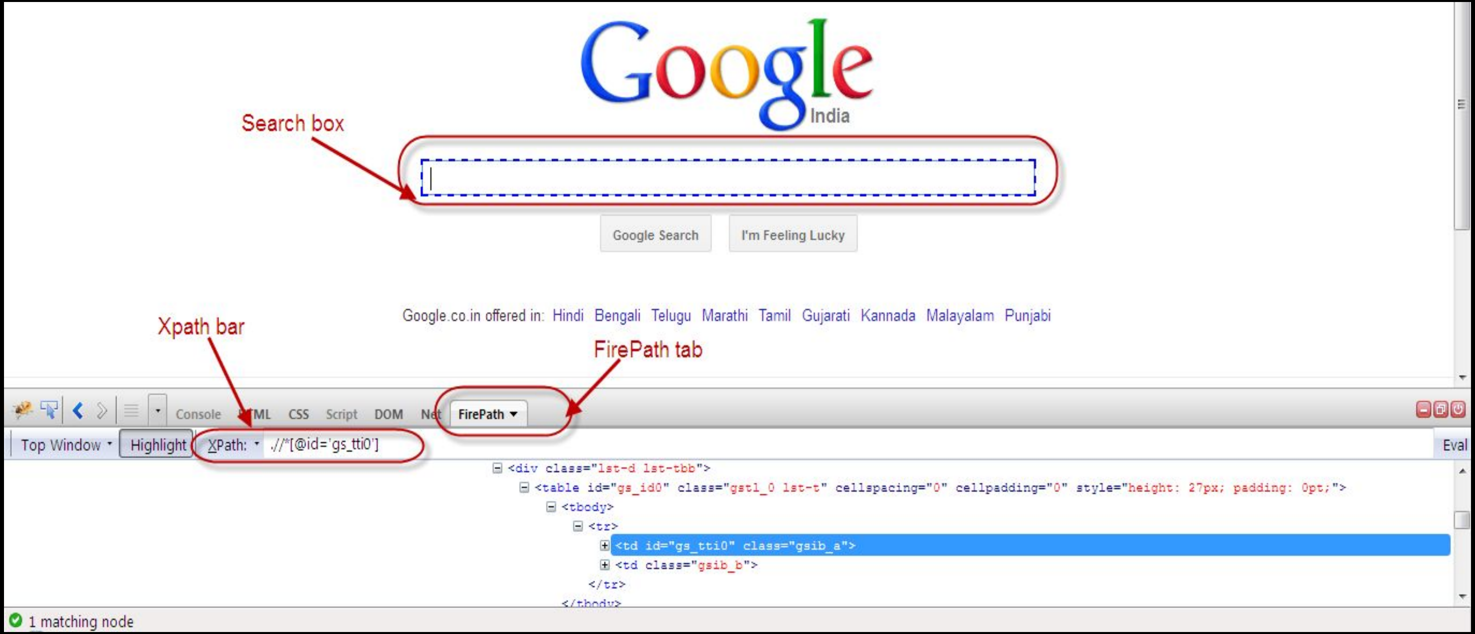
```
mydict = {'searchTerm': term}
qstr = urlencode(mydict)
```

Open the URL and wait for 10 seconds for everything to load properly

```
baseurl = "http://localhost/player.html?"
mydriver.get(baseurl + qstr)
sleep(10)
```

Selenium – finding the Xpath

Find the Xpath of an element on a website



Search box

Xpath bar

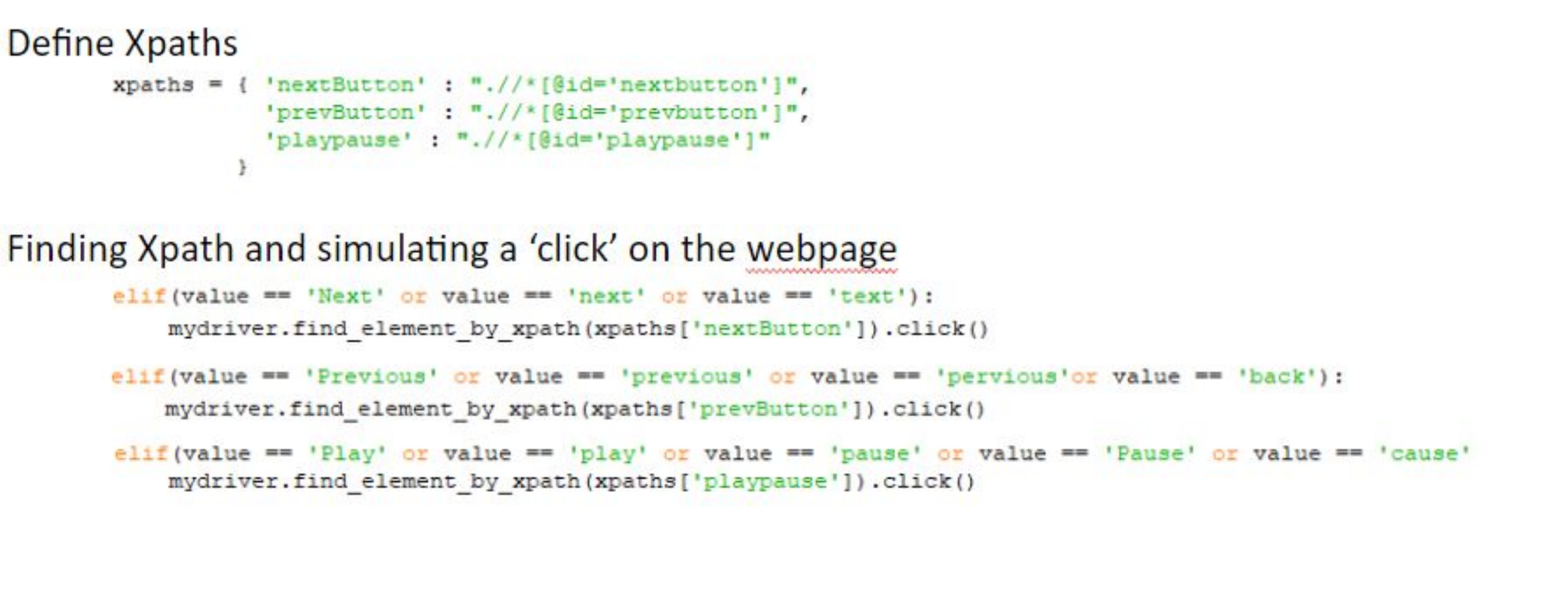
FirePath tab

XPath: .//*[@id='gs_tti0']

```
⊟ <div class="lst-d lst-tbb">
  ⊟ <table id="gs_id0" class="gstl_0 lst-t" cellspacing="0" cellpadding="0" style="height: 27px; padding: 0pt;">
    ⊟ <tbody>
      ⊟ <tr>
        ⊞ <td id="gs_tti0" class="gsib_a">
        ⊞ <td class="gsib_b">
        </tr>
      </tbody>
```

✓ 1 matching node

Selenium – finding the Xpath (Cont’d)



Define Xpaths

```
xpaths = { 'nextButton' : ".//*[@id='nextbutton']",
           'prevButton' : ".//*[@id='prevbutton']",
           'playpause' : ".//*[@id='playpause']"
         }
```

Finding Xpath and simulating a ‘click’ on the webpage

```
elif(value == 'Next' or value == 'next' or value == 'text'):
    mydriver.find_element_by_xpath(xpaths['nextButton']).click()

elif(value == 'Previous' or value == 'previous' or value == 'pervious'or value == 'back'):
    mydriver.find_element_by_xpath(xpaths['prevButton']).click()

elif(value == 'Play' or value == 'play' or value == 'pause' or value == 'Pause' or value == 'cause'
    mydriver.find_element_by_xpath(xpaths['playpause']).click()
```

COMMUNITY
INNOVATION
WORKSPACE

THANK YOU

TAG US ON OUR SOCIAL MEDIA

The Assembly

@MakeSmartThings

MakeSmartThings

The Assembly

تركيب
THEASSEMBLY
MAKE | SMART | THINGS